Templates and Item Lists

PASCAL

```
PROGRAM Test_Templates (input, output);

[HIDDEN] TYPE
      $BYTE = [BYTE] -128..127;
      $WORD = [WORD] -32768..32767;
      $QUAD = [QUAD,UNSAFE] RECORD
            L0:UNSIGNED; L1:INTEGER; END;
      $OCTA = [OCTA,UNSAFE] RECORD
            L0,L1,L2:WORD; L3:INTEGER; END;
      $UBYTE = [BYTE] 0..255;
      $UWORD = [WORD] 0..65535;
      $UQUAD = [QUAD,UNSAFE] RECORD
            L0,L1:UNSIGNED; END;
      $UOCTA = [OCTA,UNSAFE] RECORD
            L0,L1,L2,L3:UNSIGNED; END;
      $PACKED_DEC = [BIT(4),UNSAFE] 0..15;
      $DEFTYP = [UNSAFE] INTEGER;
      $DEFPTR = [UNSAFE] ^$DEFTYP;

[ASYNCHRONOUS,EXTERNAL(SYS$GETMSG)] FUNCTION $GETMSG (
%IMMED MSGID : UNSIGNED;
VAR MSGLEN : [VOLATILE] $UWORD;
VAR BUFADR : [CLASS_S,VOLATILE] PACKED ARRAY [$l3..$u3:INTEGER]
OF CHAR;
%IMMED FLAGS : UNSIGNED := %IMMED 15;
VAR OUTADR : [VOLATILE] UNSIGNED := %IMMED 0) : INTEGER;
EXTERNAL;

[ASYNCHRONOUS,EXTERNAL(SYS$SNDJBCW)] FUNCTION $SNDJBCW (
%IMMED EFN : UNSIGNED := %IMMED 0;
%IMMED FUNC : INTEGER;
%IMMED NULLARG : UNSIGNED := %IMMED 0;
%REF ITMLST : [UNSAFE] ARRAY [$l4..$u4:INTEGER] OF $UBYTE :=
%IMMED 0;
VAR IOSB : [VOLATILE] $UQUAD := %IMMED 0;
%IMMED [UNBOUND, ASYNCHRONOUS] PROCEDURE ASTADR := %IMMED 0;
%IMMED ASTPRM : UNSIGNED := %IMMED 0) : INTEGER; EXTERNAL;


CONST
SJC$_ENTRY_NUMBER_OUTPUT = 31;
SJC$_FILE_SPECIFICATION = 42;
```

C++

```
/** @file **/
//
//        Sector7 USA Inc, VMS PASCAL to C++, Version 7.2
//        Build:  8.000 2017/09/16 23:06:29 (GMT)
//        VX/RT (c) 1985-2017 Sector7 USA LLC.
//        Translated:  2018-07-05 08:23:13 CDT
//        Input File:  QA_Templates.pas
//        Output File: QA_Templates.cxx
//
//

#include "VxPascal.hxx"

typedef char $BYTE ;

typedef int16_t $WORD ;

typedef _QUAD $QUAD;

typedef struct $OCTA  {
    //! @ingroup TEST_TEMPLATES
/* Warning VMS ATTRIBUTE [OCTA(OCTA)] Not Supported */

// vxpascal: QA_Templates.pas, line 9:
// * VMS ATTRIBUTE [OCTA(OCTA)] Not Supported
    InitializeRecord($OCTA)
    int16_t L0 ;
    int16_t L1 ;
    int16_t L2 ;
    int32_t L3 ;
} $OCTA;

typedef uchar_t $UBYTE ;

typedef uint16_t $UWORD ;

typedef _QUAD $UQUAD;

typedef struct $UOCTA  {
/* Warning VMS ATTRIBUTE [OCTA(OCTA)] Not Supported */
// vxpascal: QA_Templates.pas, line 15:
// * VMS ATTRIBUTE [OCTA(OCTA)] Not Supported
```

```
SJC$_QUEUE = 134;
SJC$_USERNAME = 159;
SJC$_JOB_STATUS_OUTPUT = 88;
SJC$_DELETE_FILE = 24;
SJC$_ENTER_FILE = 19;
SJC$_LOG_SPECIFICATION = 98;
SJC$_JOB_NAME = 79;



FUNCTION  fdSubmitCommandTask( vfile_to_run : [VOLATILE]
VARYING[L1] OF CHAR := ' ';
vQueueName           : PACKED ARRAY [s2..f2:INTEGER] OF CHAR := '
';
VAR Entry_Number     : INTEGER) : BOOLEAN;


TYPE
Item_List_Cell = RECORD CASE INTEGER OF
1: (
IsTest        : BOOLEAN;
Buffer_Length : [WORD] 0..65535;
Item_Code     : [WORD] 0..65535;
Buffer_Addr   : UNSIGNED;
Return_Addr   : UNSIGNED

);
2:(
Terminator    : UNSIGNED
);
END;

Item_List_Template(Count:INTEGER) = ARRAY [1..Count] OF
Item_List_Cell;

IOSB_def = RECORD
STATUS : INTEGER;
Reserved   : INTEGER;
END;

VAR
Item_List        : Item_List_Template(9);
sys_stat         : INTEGER;
fig                                : INTEGER;
dec              : integer;
IOSB             : IOSB_def;
file_to_run      : [volatile] VARYING [50] OF CHAR;
log_file         : [volatile] VARYING [50] OF CHAR;
queue            : [volatile] VARYING [50] OF CHAR;
username         : [volatile] VARYING [50] OF CHAR;
jobname          : [volatile] VARYING [20] OF CHAR;
```

```
    InitializeRecord($UOCTA)
    uint32_t L0 ;
    uint32_t L1 ;
    uint32_t L2 ;
    uint32_t L3 ;
} $UOCTA;

typedef uchar_t $PACKED_DEC ;

typedef int32_t $DEFTYP ;

typedef int32_t& $DEFPTR ;


extern int32_t sys$GETMSG(
/* |PR|                    */  uint32_t MSGID ,
/* |VR|VL|                 */  $UWORD& MSGLEN ,
/* |VR|VL|%C_S|            */  _String& BUFADR ,
/* |PR|                    */  uint32_t FLAGS  = 15,
/* |VR|VL|                 */  uint32_t& OUTADR  = *(uint32_t *)
0) ;


extern int32_t (
/* |PR|                    */  uint32_t EFN  = 0,
/* |PR|                    */  int32_t FUNC  = 0,
/* |PR|                    */  uint32_t NULLARG  = 0,
/* |PR|US|%REF|            */  _ArrayRef<uchar_t> ITMLST  = 0,
/* |VR|VL|                 */  $UQUAD& IOSB  = *($UQUAD *) 0,
/* |PR|BO|AS|              */  uint32_t ASTADR  = 0,
/* |PR|                    */  uint32_t ASTPRM  = 0) ;


static const int32_t  SJC$_ENTRY_NUMBER_OUTPUT    = 31;
static const int32_t  SJC$_FILE_SPECIFICATION     = 42;
static const int32_t  SJC$_QUEUE                  = 134;
static const int32_t  SJC$_USERNAME               = 159;
static const int32_t  SJC$_JOB_STATUS_OUTPUT      = 88;
static const int32_t  SJC$_DELETE_FILE            = 24;
static const int32_t  SJC$_ENTER_FILE             = 19;
static const int32_t  SJC$_LOG_SPECIFICATION      = 98;
static const int32_t  SJC$_JOB_NAME               = 79;

typedef struct _r1_Item_List_Cell  {
    InitializeRecord(_r1_Item_List_Cell)
    union {
        struct {
            bool IsTest ;
            uint16_t Buffer_Length ;
            uint16_t Item_Code ;
            uint32_t Buffer_Addr ;
            uint32_t Return_Addr ;
```

```
jbc_status_line  : [volatile] VARYING [132] OF CHAR;
sys$message      : VARYING [256] OF CHAR;
i                : INTEGER;

BEGIN

Item_List[7].Buffer_Length          :=
SIZE(JBC_Status_Line.Body);
Item_List[7].Item_Code              := SJC$_Job_Status_Output;
Item_List[7].Buffer_Addr            :=
IADDRESS(JBC_Status_Line.Body);
Item_List[7].Return_Addr            :=
IADDRESS(JBC_Status_Line.Length);

File_to_run := 'COM_DIR:RUNTASK_' + vfile_to_run + '.COM';
Item_List[1].Buffer_Length          := LENGTH(File_To_run);
Item_List[1].Item_Code              := SJC$_FILE_SPECIFICATION;
Item_List[1].Buffer_Addr            :=
IADDRESS(File_To_run.Body);
Item_List[1].Return_Addr            := 0;

queue      := vQueueName;
Item_List[2].Buffer_Length          := LENGTH(Queue);
Item_List[2].Item_Code              := SJC$_QUEUE;
Item_List[2].Buffer_Addr            := IADDRESS(Queue.Body);
Item_List[2].Return_Addr            := 0;

log_file := 'LOG_DIR:TENGINE_' + vfile_to_run + '.LOG';

Item_List[3].Buffer_Length          := LENGTH(log_file);
Item_List[3].Item_Code              := SJC$_LOG_SPECIFICATION;
Item_List[3].Buffer_Addr            := IADDRESS(log_file.Body);
Item_List[3].Return_Addr            := 0;

username          := 'MHSMAINT';
Item_List[4].Buffer_Length          := LENGTH(username);
Item_List[4].Item_Code              := SJC$_USERNAME;
Item_List[4].Buffer_Addr            := IADDRESS(username.Body);
Item_List[4].Return_Addr            := 0;

jobname := vfile_to_run;

Item_List[5].Buffer_Length          := LENGTH(jobname);
Item_List[5].Item_Code              := SJC$_JOB_NAME;
Item_List[5].Buffer_Addr            := IADDRESS(jobname.Body);
Item_List[5].Return_Addr            := 0;

Item_List[6].Buffer_Length          := 4;
Item_List[6].Item_Code              := SJC$_ENTRY_NUMBER_OUTPUT;
Item_List[6].Buffer_Addr            := IADDRESS(ENTRY_NUMBER);
Item_List[6].Return_Addr            := 0;
```

```
        } ;
        uint32_t Terminator ;
    };
} _r1_Item_List_Cell;

template < int32_t _Count=0 >
class Item_List_Template {
    public:
        static const int32_t  Count = _Count;
        _r1_Item_List_Cell  data [Count];
};


static bool  fdSubmitCommandTask(
/* |PR|VL|                */  _String vfile_to_run ,
/* |PR|                   */  _String vQueueName ,
/* |VR|                   */  int32_t& Entry_Number )  {

    typedef struct IOSB_def  {
        InitializeRecord(IOSB_def)
        int32_t STATUS ;
        int32_t Reserved ;
    } IOSB_def;

    _VaryingString<50>  file_to_run;
    _VaryingString<50>  log_file;
    _VaryingString<50>  queue;
    _VaryingString<50>  username;
    _VaryingString<20>  jobname;
    _VaryingString<132>  jbc_status_line;
    _VaryingString<256>  sys$message;
    bool  _fdSubmitCommandTask_retv ;
    _ArrayOf<1,9,_r1_Item_List_Cell>  Item_List ;
    int32_t  sys_stat ;
    int32_t  fig ;
    int32_t  DEC ;
    IOSB_def  IOSB ;
    int32_t  i ;

    Item_List.data[7].Buffer_Length =
sizeof(jbc_status_line.Body);
    Item_List.data[7].Item_Code = SJC$_JOB_STATUS_OUTPUT;
    Item_List.data[7].Buffer_Addr = Address(jbc_status_line);
    Item_List.data[7].Return_Addr =
Address(&jbc_status_line._Length());
    file_to_run  =  "COM_DIR:RUNTASK_" + vfile_to_run + ".COM";
    Item_List.data[1].Buffer_Length = Length(file_to_run);
    Item_List.data[1].Item_Code = SJC$_FILE_SPECIFICATION;
    Item_List.data[1].Buffer_Addr = Address(file_to_run);
    Item_List.data[1].Return_Addr = 0;
    queue  =  vQueueName;
    Item_List.data[2].Buffer_Length = Length(queue);
```

```
Item_List[7].Buffer_Length          :=
SIZE(JBC_Status_Line.Body);
Item_List[7].Item_Code              := SJC$_Job_Status_Output;
Item_List[7].Buffer_Addr            :=
IADDRESS(JBC_Status_Line.Body);
Item_List[7].Return_Addr            :=
IADDRESS(JBC_Status_Line.Length);

Item_List[8].Buffer_Length          := 0;
Item_List[8].Item_Code              := SJC$_DELETE_FILE;
Item_List[8].Buffer_Addr            := 0;
Item_List[8].Return_Addr            := 0;

Item_List[9].Terminator  := 0;

SYS_STAT := $SNDJBCW ( FUNC   := %IMMED SJC$_ENTER_FILE,
ITMLST := ITEM_LIST,
IOSB   := IOSB);
IF (NOT ODD(SYS_STAT)) OR (NOT ODD(IOSB.STATUS)) THEN
      BEGIN
            $GETMSG(msgid := IOSB.STATUS,
            msglen := sys$message.length,
            bufadr := sys$message.body);
            WRITEV(sys$message, sys_stat:fig:dec);
            WRITELN (sys$message);
            WRITELN ('### Fail to submit the task job =
',file_to_run);
            WRITELN ('### SNDJBCW err,SYS_STAT: ',SYS_STAT:0,'
IOSB.STATUS: ',IOSB.STATUS:0,' fdSubmitCommandTask');
            WRITELN (sys$message);
            fdSubmitCommandTask := FALSE;
      END
ELSE
      BEGIN
            WRITELN (jbc_status_line);
            fdSubmitCommandTask := TRUE;
      END;
END;




FUNCTION  testTemplate  : BOOLEAN;
      BEGIN
            testTemplate:=true;
      END;

CONST
            element_offset = 2 * SIZE( INTEGER );
            c_index_size = 1000;

TYPE
```

```
    Item_List.data[2].Item_Code = SJC$_QUEUE;
    Item_List.data[2].Buffer_Addr = Address(queue);
    Item_List.data[2].Return_Addr = 0;
    log_file  =  "LOG_DIR:TENGINE_" + vfile_to_run + ".LOG";
    Item_List.data[3].Buffer_Length = Length(log_file);
    Item_List.data[3].Item_Code = SJC$_LOG_SPECIFICATION;
    Item_List.data[3].Buffer_Addr = Address(log_file);
    Item_List.data[3].Return_Addr = 0;
    username  =  "MHSMAINT";
    Item_List.data[4].Buffer_Length = Length(username);
    Item_List.data[4].Item_Code = SJC$_USERNAME;
    Item_List.data[4].Buffer_Addr = Address(username);
    Item_List.data[4].Return_Addr = 0;
    jobname  =  vfile_to_run;
    Item_List.data[5].Buffer_Length = Length(jobname);
    Item_List.data[5].Item_Code = SJC$_JOB_NAME;
    Item_List.data[5].Buffer_Addr = Address(jobname);
    Item_List.data[5].Return_Addr = 0;
    Item_List.data[6].Buffer_Length = 4;
    Item_List.data[6].Item_Code = SJC$_ENTRY_NUMBER_OUTPUT;
    Item_List.data[6].Buffer_Addr = Address(&Entry_Number);
    Item_List.data[6].Return_Addr = 0;
    Item_List.data[7].Buffer_Length =
sizeof(jbc_status_line.Body);
    Item_List.data[7].Item_Code = SJC$_JOB_STATUS_OUTPUT;
    Item_List.data[7].Buffer_Addr = Address(jbc_status_line);
    Item_List.data[7].Return_Addr =
Address(&jbc_status_line._Length());
    Item_List.data[8].Buffer_Length = 0;
    Item_List.data[8].Item_Code = SJC$_DELETE_FILE;
    Item_List.data[8].Buffer_Addr = 0;
    Item_List.data[8].Return_Addr = 0;
    Item_List.data[9].Terminator = 0;
    sys_stat = SJC$_ENTRY_NUMBER_OUTPUT(
        0,
        SJC$_ENTER_FILE,
        0,
        Item_List.data,
        IOSB);
    if ((!Odd(sys_stat)) || (!Odd(IOSB.STATUS))) {
            sys$GETMSG(
                        (uint32_t)IOSB.STATUS,
                        sys$message._Length(),
                        sys$message);
        Writev(E_CONTINUE, "%D%*.*i", (DXX *)(sys$message), fig,
DEC, sys_stat);
        Writeln(OUTPUT, E_CONTINUE, "%D", (DXX *)(sys$message));
        Writeln(OUTPUT,
                E_CONTINUE,
                "%s%D",
                "### Fail to submit the task job = ",
                (DXX *)(file_to_run));
```

```
            element_ptr_t = ^element_t;
            element_t( upper_bound: INTEGER ) = PACKED RECORD
            prev,
            next    : element_ptr_t;
            element : PACKED ARRAY [1..upper_bound ] OF CHAR;
            END;

            index_ptr_t = ^index_t;
            index_t = RECORD
             element_ptr  : PACKED ARRAY [1..c_index_size] OF
element_ptr_t;
            END;

TYPE
listheader_def = ^listheader_type VALUE NIL;
listheader_type =
RECORD
first,
last    : [LONG, UNSAFE] element_ptr_t VALUE NIL;
datasize,
no_of_elements,
reserve : INTEGER VALUE ZERO;
index   : index_ptr_t VALUE NIL;
END;

END.
```

```
        Writeln(OUTPUT,
                E_CONTINUE,
                "%s%*i%s%*i%s",
                "### SNDJBCW err,SYS_STAT: ",
                0,
                sys_stat,
                " IOSB.STATUS: ",
                0,
                IOSB.STATUS,
                " fdSubmitCommandTask");
        Writeln(OUTPUT, E_CONTINUE, "%D", (DXX *)(sys$message));
        _fdSubmitCommandTask_retv = false;
    } else  {
            Writeln(OUTPUT, E_CONTINUE, "%D", (DXX
*)(jbc_status_line));
        _fdSubmitCommandTask_retv = true;
    }
    return _fdSubmitCommandTask_retv;

}


static bool  testTemplate(void)  {
    bool  _testTemplate_retv ;

    _testTemplate_retv = true;
    return _testTemplate_retv;

}


static const int32_t  element_offset              =
(sizeof(int32_t) * 2);
static const int32_t  c_index_size                = 1000;

typedef struct _AnyRef * element_ptr_t ;
template < int32_t _upper_bound=0 >
class element_t {
    public:
        static const int32_t  upper_bound = _upper_bound;
        element_ptr_t prev ;
        element_ptr_t next ;
        _FixedString<1,upper_bound> element ;
};

typedef struct index_t * index_ptr_t ;

typedef struct index_t  {
    InitializeRecord(index_t)
    _ArrayOf<1,c_index_size,struct _AnyRef *> element_ptr ;
} index_t;
```

```
typedef struct listheader_type * listheader_def ;

typedef struct listheader_type  {
    listheader_type()
    {
         first = 0;
         datasize = 0;
         index = 0;
    };
    InitializeRecord(listheader_type)
    struct element_ptr_t first ;
    struct element_ptr_t last ;
    int32_t datasize ;
    int32_t no_of_elements ;
    int32_t reserve ;
    struct index_ptr_t index ;
} listheader_type;


/* PROGRAM SELF CHECK END */int main(int argc, const char
*argv[])
{
        vxrt_exit(0);
}




class Test_TemplatesToDo : public ToDo {
    public:
        Test_TemplatesToDo() : ToDo("Test_Templates") {
OnBegin(); };
        ~Test_TemplatesToDo() { OnEnd(); };
        void OnBegin();
        void OnEnd();
};
static Test_TemplatesToDo todo;
void Test_TemplatesToDo::OnBegin() {
}
void Test_TemplatesToDo::OnEnd() {
}
```